Hochschule
Ravensburg-Weingarten
Technik | Wirtschaft | Sozialwesen

DEPARTAMENTO DE
INGENIERÍA ELÉCTRICA
Y CIENCIAS DE LA COMPUTACIÓN

PROYECTO FIN DE CARRERA

CONTROL Y PROGRAMACIÓN DE UN ROBOT INDUSTRIAL CON MICROSOFT KINECT

AUTOR: **Pablo Pérez Tejeda**

PROFESOR COTUTOR: **Prof. Basam Musleh Lancis**

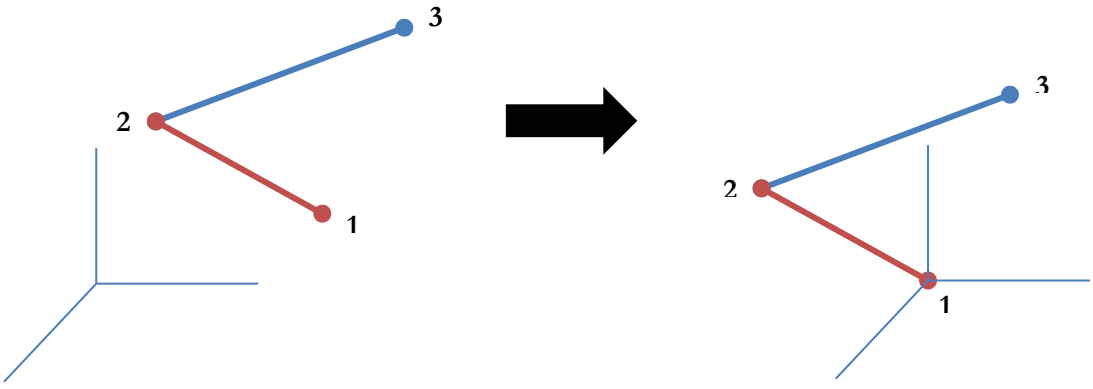
FECHA DE ENTREGA: **31 de Agosto de 2013**

Resumen del proyecto

1. Introducción

Un robot industrial puede ser programado de múltiples maneras, que van desde la escritura de código en lenguajes de alto nivel como RAPID, hasta el uso de entornos de programación más o menos gráficos, como Robotstudio (para robots de la firma ABB). En los últimos años se ha realizado un considerable esfuerzo en la búsqueda de interfaces de programación más intuitivas mediante el uso de cámaras y el empleo de algoritmos de reconocimiento de gestos. En este contexto encontramos el sensor Kinect de Microsoft, que surgió como un periférico de la consola Xbox 360, pero que ha ido ganando terreno en aplicaciones de reconocimiento de imágenes y visión por computador.

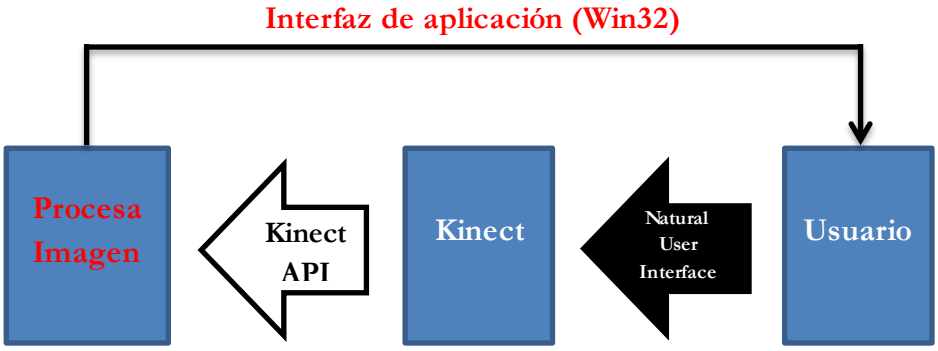
El objetivo del proyecto es el diseño e implementación del software encargado de identificar, localizar y representar tres objetos en tiempo real, basándose en sus características de color, a través del uso del sensor Kinect. Este sistema debe servir como interfaz para la programación de un robot (a partir de las coordenadas de los tres objetos se definirían la posición y orientación de una hipotética herramienta respecto al sistema de referencia del robot), aunque este paso no ha sido llevado a cabo en este proyecto.



El sensor debe ser programado en lenguaje C++ usando el SDK de Kinect para Windows, y la aplicación desarrollada debe incluir una interfaz gráfica de usuario (GUI, por sus siglas en inglés), para lo cual se utiliza la interfaz de usuario de aplicaciones de escritorio para Windows (Windows Desktop API). El entorno de desarrollo usado es Visual Studio 2012. Para los algoritmos de visión por computador se usan las librerías de OpenCV.

La entrada de datos a través de la cámara se produce a través del hardware del sensor Kinect (sólo los sensores RGB y de profundidad son utilizados en este proyecto) y de una tubería de software dedicado junto con unas librerías específicas incluidas en la interfaz de programación de aplicación (Kinect API) de Kinect para Windows. Los algoritmos utilizados para el reconocimiento de gestos por parte de la Kinect reciben el nombre de Interfaz Natural de Usuario (en inglés, Natural User Interface).

El siguiente esquema resalta en rojo las partes que se llevan a cabo en este proyecto.



La API de Kinect permite acceder directamente a los datos proporcionados por el sensor a través de unos registros denominados "streams", que se actualizan continuamente y pueden ser solicitados para su lectura por parte del programa. Existen streams para los datos de la cámara RGB, para el sensor de profundidad así como para el resto de sensores de la Kinect. Estos streams presentan una tasa de actualización de hasta 30 FPS (frames per second).

Como resumen del trabajo que se lleva a cabo en este proyecto:

- Se usa la funcionalidad básica de la Kinect (sensores de color y profundidad) para capturar información de la escena
- Esta información es procesada a través de algoritmos tomados de las librerías de OpenCV para lograr el reconocimiento de los objetos en la escena
- El resultado del reconocimiento (posición de los objetos en la escena) es mostrado por medio de la pantalla al usuario, de este modo se puede comprobar si el sistema funciona de la manera deseada
- En un paso adicional (no desarrollado en este proyecto) se pasaría de la simulación, al control real de un robot, para lo cual debería enviarse al robot los datos de posición y orientación deseados de su herramienta de trabajo, a partir de la información obtenida de la posición de los objetos reconocidos

2. Introducción al algoritmo de reconocimiento

El reconocimiento de objetos en este caso tan simple se consigue mediante un único paso de segmentación.

La segmentación es un concepto de visión por computador basado en la división de una imagen en grupos de píxeles o superpíxeles con el objetivo de simplificar o cambiar la representación actual en algo que es más significativo o fácil de analizar. El resultado de la segmentación es un conjunto de segmentos o regiones que colectivamente cubren toda la imagen. Cada uno de los píxeles en una región es similar con respecto a alguna característica o propiedad computada, y las regiones adyacentes son significativamente diferentes con respecto a la misma característica.

En el caso que nos incumbe, la segmentación se utiliza para separar los tres objetos a reconocer del resto de la imagen, y también entre sí. Para lograrla se aplican dos pasos.

El primer paso comprende la discriminación de píxeles con respecto a diferencias de tonalidad de color (hue, en inglés). La tonalidad o tono de color es una de las propiedades principales del color, junto con otras propiedades asociadas a la apariencia del color, como contraste, cromaticidad, saturación, luminosidad o brillo. El tono se define técnicamente como el grado en el que un estímulo puede ser descrito como similar o diferente de otros estímulo descritos como rojo, verde, azul y amarillo. A diferencia de otros algoritmos que usan información geométrica de la escena (formas, contornos…) la segmentación de color está basada únicamente en el histograma de color de la imagen y por tanto no tiene en cuenta la posición relativa de los píxeles en ella. Esto es fruto de una deficiencia en el algoritmo de segmentación que se corrige mediante el segundo paso.

El segundo paso consiste en la separación de la parte significativa de la imagen (objetos móviles o en el primer plano) de la parte no significativa, o distorsionadora (el fondo de la imagen y/o aquellas partes de la escena que permanecen fijas y por tanto no tienen interés). Para lograr esta separación se utiliza un algoritmo que permite distinguir los píxeles que permanecen inmóviles del resto de píxeles móviles, por medio de un modelo que describe el "fondo" de la imagen, y que es actualizado en cada nueva imagen capturada por la cámara. Un determinado píxel se compara con el "modelo", y mediante unos ciertos parámetros, se determina si pertenece al fondo o a los objetos en primer plano. De esta manera, el fondo de la imagen puede ser eliminado.

3. Segmentación de color

Los datos de color pueden ser recogidos del sensor Kinect en diferentes resoluciones y formatos. La resolución escogida es 640x480 píxeles, que no es la máxima que ofrece el sensor, pero permite una tasa de procesamiento adecuada. El formato escogido en esta aplicación es RGB, por el que los datos son adquiridos en mapas de bits de 32-bit y formato X8R8G8B8 (es decir, 8 bit para cada componente básica del color más un byte extra que almacena otro tipo de información). Cada imagen puede ser estudiada por tanto como un conjunto de tres matrices de dimensión 2 conteniendo la información de las tres componentes de cada píxel, rojo, verde y azul, en valores comprendidos entre 0 y 255.

El modelo RGB es un modelo aditivo de color en el cual se superponen rayos de luz roja, verde y azul de diferentes maneras para lograr un amplio espectro de colores. El principal propósito de este modelo es el sensado, representación y visualización de imágenes en sistemas electrónicos, como televisores y computadoras, pero no es adecuado para la descripción cualitativa del color, por ser un modelo poco intuitivo. Descripciones en términos de tono/luminosidad/cromaticidad o tono/luminosidad/saturación son más relevantes.

Los espacios HSV y HSL (por sus siglas en inglés) atienden a estas consideraciones. Estos espacios o modelos tienen más en cuenta cómo los colores son organizados y conceptualizados en términos de la visión humana por medio de atributos como tono, luminosidad y cromaticidad.

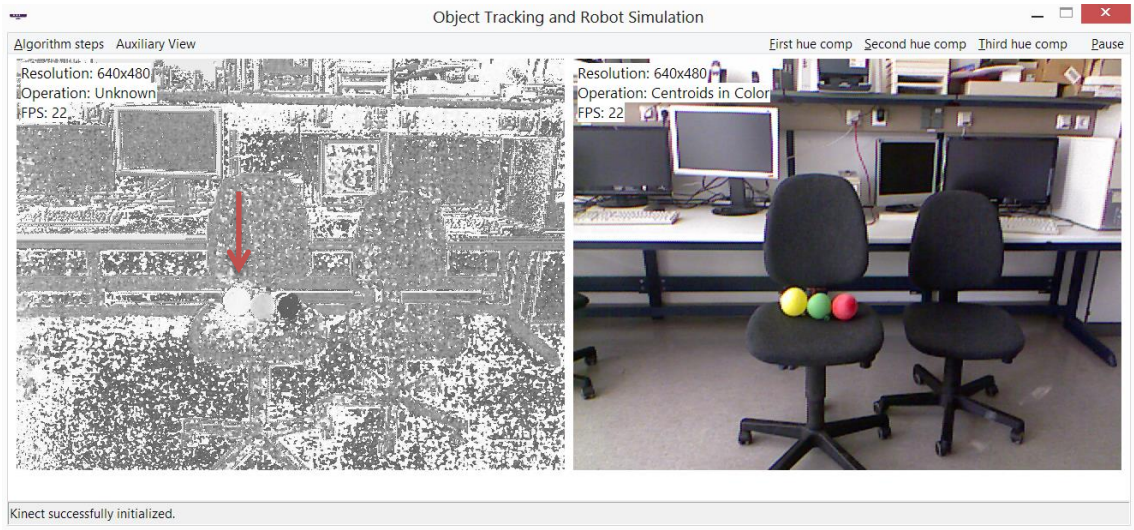
Estos modelos pueden ser derivados mediante consideraciones geométricas del modelo RGB, aunque estas consideraciones no se incluyen en este resumen. De esta manera podemos obtener la representación de la componente de tono en un rango que varía de 0 a 255 (tamaño de un byte), o de 0 a 127 en su representación en las librerías de OpenCV.

La función utilizada de las librerías de OpenCV para la conversión del espacio de color es **cvtColor**, llamada con el parámetro CV_BGR2HSV. Una vez realizada la conversión, la imagen pasa a ser una imagen de un solo canal, en lugar de los cuatro canales iniciales, con lo que la imagen se muestra en valores de grises. Dichos valores representan el tono de cada píxel en la imagen. Los píxeles con el mismo valor de gris corresponderán también a los píxeles de la escena real que tienen la misma propiedad de color.

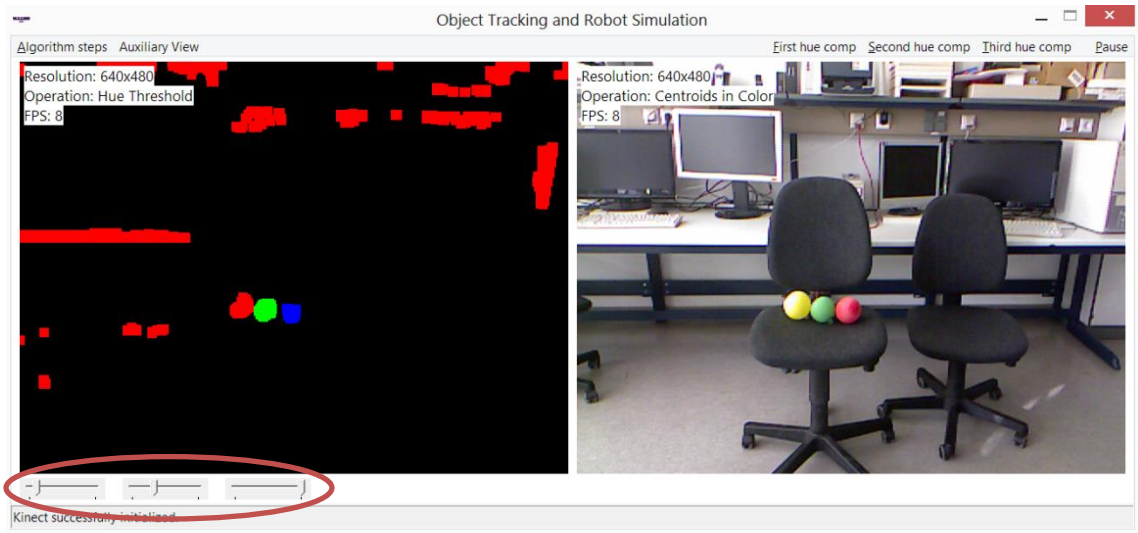
Para realizar la segmentación de color de la imagen el siguiente paso es aplicar un umbralizado a la imagen, en el que nos quedamos solo con los pixeles cuyo valor de gris se encuentre dentro de un rango que nosotros definimos. Así, los píxeles cercanos al color que deseamos encontrar, tomarán un valor de 1 y el resto tomarán un valor de 0. A esta operación de conversión de un conjunto de valores discretos a sólo dos valores se le denomina binarizado, y es una consecuencia natural de la operación de umbralizado.

La operación de segmentación de color se puede realizar con tantos umbrales como deseemos, en este caso umbralizaremos buscando los valores de tono de cada uno de los tres objetos.

En la siguiente captura del programa, se muestra el paso previo a la umbralización para el primer objeto (tono buscado: amarillo), en el que se discrimina la distancia al tono de color buscado de cada píxel. Los píxeles con una distancia menor aparecen como más claros, y los píxeles con una distancia mayor, como más oscuros.



La siguiente captura muestra el resultado de la umbralización aplicada a cada objeto y combinada en una única imagen, devolviendo a la misma al espacio de color y mostrando un objeto en cada uno de los 3 canales (rojo, verde y amarillo). Las barras deslizantes resaltadas en la ventana son los controles utilizados para realizar el ajuste manual del programa. Este aspecto se explica en la última parte de este resumen.



4. Sustracción del fondo

La mayoría de algoritmos de sustracción de fondo se basan en la detección de objetos móviles a partir de la diferencia entre la imagen actual y una imagen de referencia, a menudo llamada "imagen de fondo" o "modelo del fondo". La sustracción de fondo se realiza casi siempre cuando la imagen es parte de un stream de video. Por otro lado, un algoritmo de sustracción de fondo robusto debe ser capaz de manejar cambios de iluminación, repentinos cambios de posición debidos a ruido y cambios de escena a largo plazo.

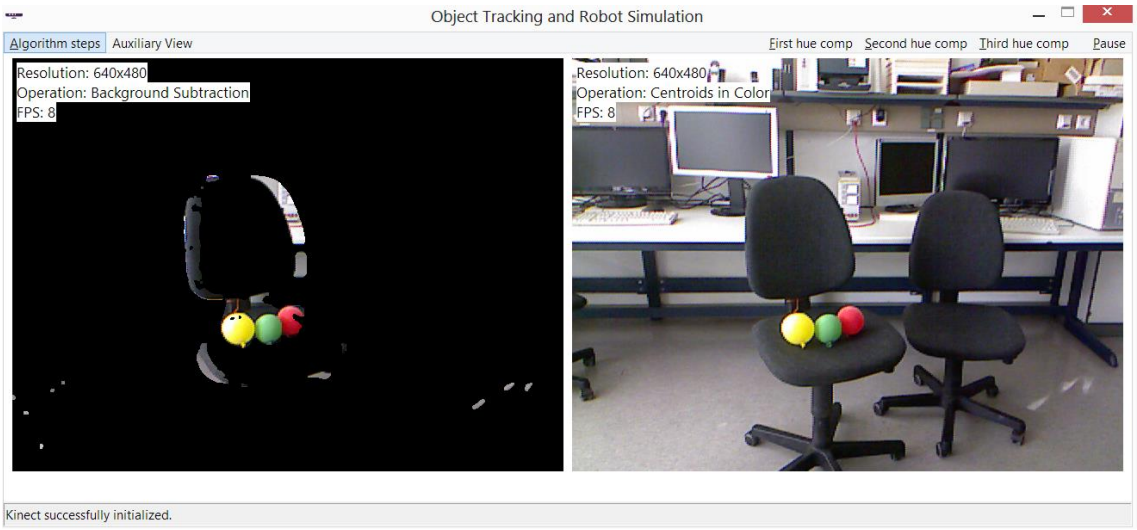
La aproximación más simple para la sustracción de fondo es usar resta de imágenes, donde la imagen actual es tomada como fondo y la distancia de cada píxel a la siguiente imagen es

comparada con un cierto umbral para discriminar la presencia de movimiento. Una aproximación más elaborada utiliza un filtro de basado en medias, para promediar una serie de imágenes precedentes y hacer un modelo más confiable del fondo. Esa es la aproximación aplicada en este proyecto.

La técnica utilizada se denomina "Modelo de mezcla de fondos". En esta técnica, se asume que la intensidad de cada píxel en el vídeo puede ser modelada como un modelo de mezcla Gaussiana. Un simple heurístico determina qué intensidades pertenecen más probablemente al fondo. Después, los píxeles que no verifican este modelo son denominados píxeles de primer plano, y son conectados usando análisis de conectividad de componentes en 2D.

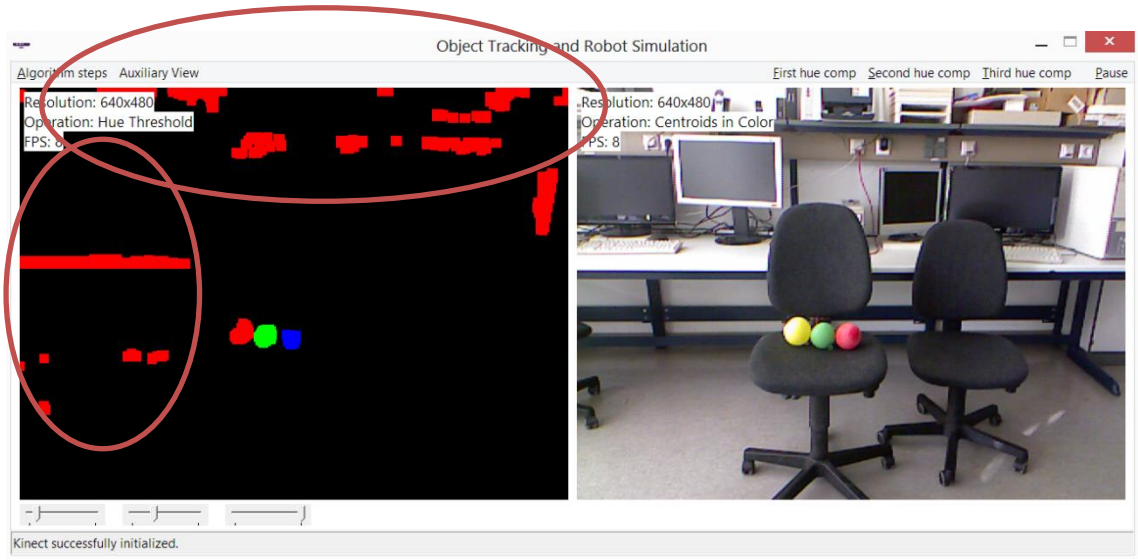
El algoritmo exacto es una mejora del modelo de mezcla de fondos, que resuelve problemas típicos de esta técnica, como tasa lenta de aprendizaje y proporciona detección de sombras. Para distinguir entre sombras móviles y objetos móviles utiliza un modelo de color que diferencia la componente cromática de la componente de iluminación, y mantiene un modelo cromático independiente de la iluminación. Esto hace el algoritmo robusto contra cambios de iluminación.

Para implementar el algoritmo de sustracción de fondo en OpenCV se utiliza la clase **BackgroundSubtractorMOG** junto con el operador **( )** de la clase. Este operador actualiza el modelo del fondo y devuelve una máscara de la imagen en primer plano. Una operación adicional de eliminación de ruido es necesaria para mejorar el algoritmo. El resultado es el que se muestra en la siguiente captura del programa.

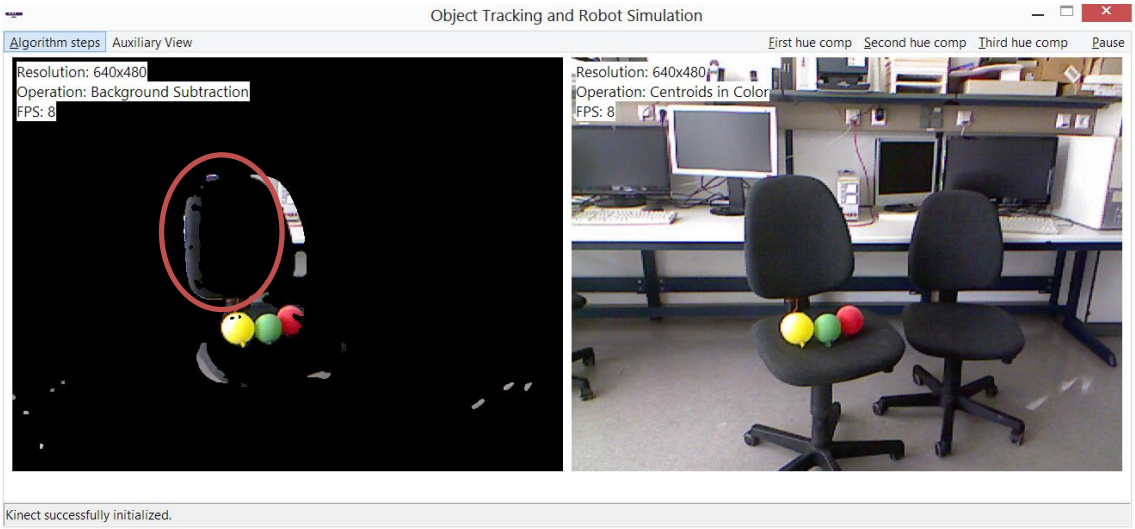


5. Efecto Combinado

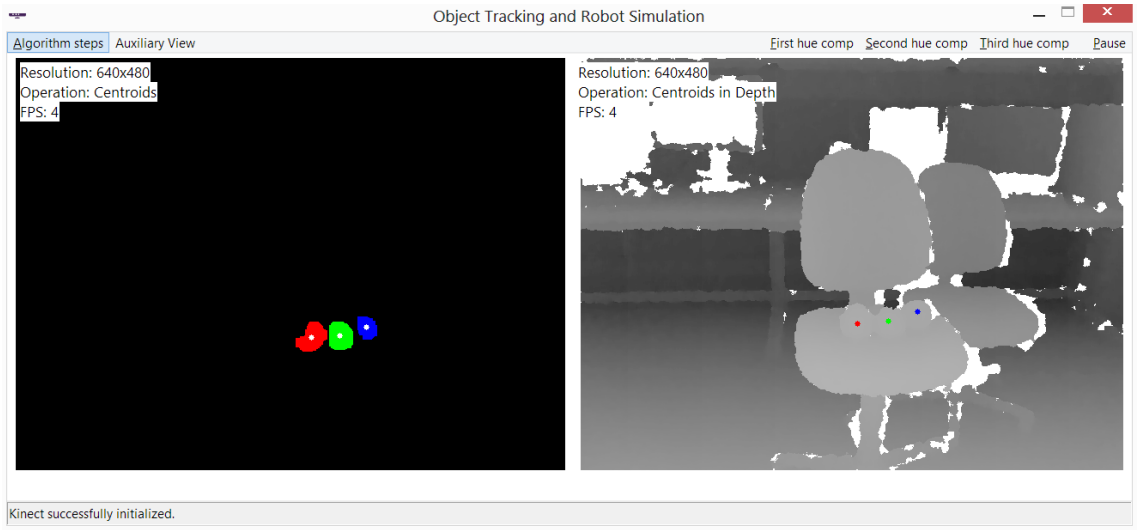
La segmentación de color en sí misma no es efectiva como algoritmo de segmentación, ya que la misma característica de color buscada en el objeto puede aparecer en el fondo, desvirtuando la etapa de umbralizado, ya que esta acepta más píxeles de los que pertenecen estrictamente al objeto.



Por otro lado, la sustracción de fondo tampoco conduce a una correcta segmentación, ya que la característica de movimiento de los objetos puede ser compartida por otros píxels en la imagen, como por ejemplo una mano que sujeta el objeto.



Sin embargo, cuando se combinan, la posibilidad de obtener una buena segmentación incrementa: cuando la segmentación de color es aplicada despues de eliminar el fondo de la imagen no aparecen interferencias del fondo y la segmentación es más efectiva, como se muestra en la última captura.

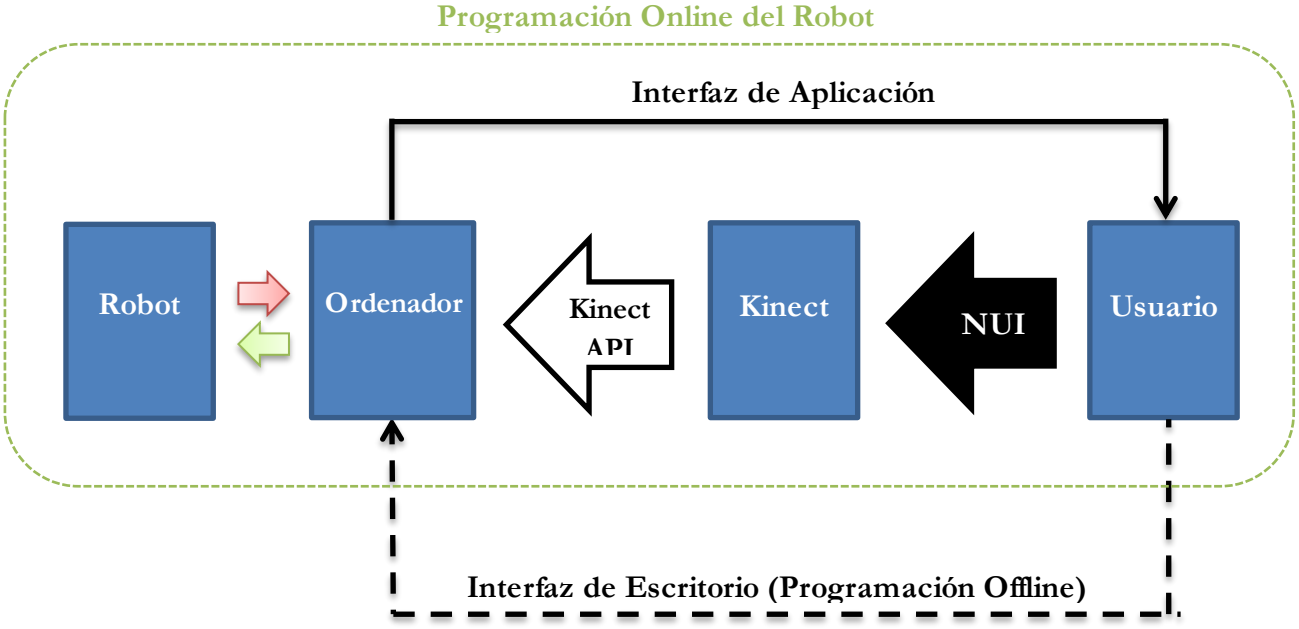


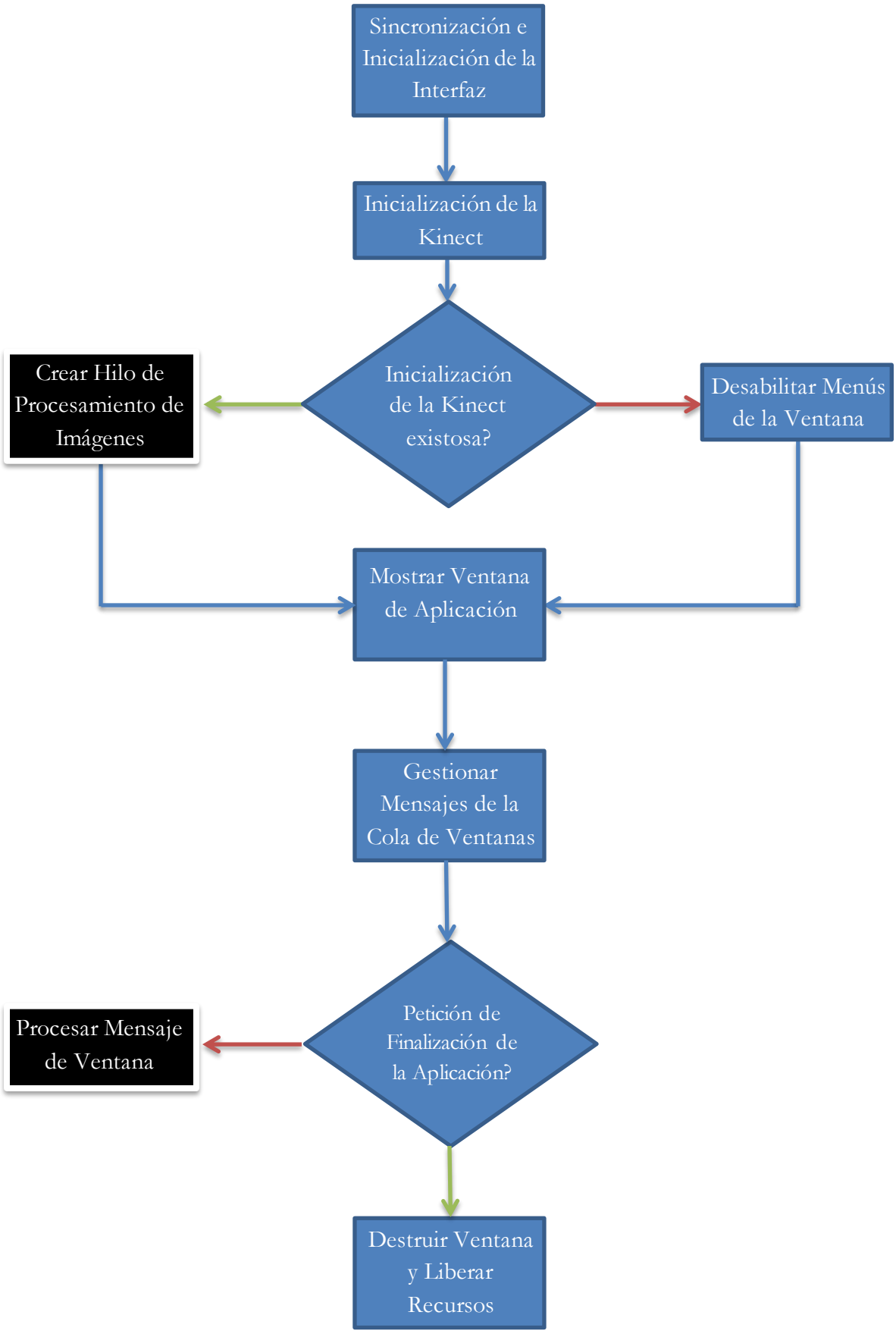
En la imagen anterior, el centro de los objetos se calcula a través del centroide de un conjunto de píxeles, que se define como la media aritmética de la posición de todos los puntos en una forma.

$$\bar{x} = \frac{M_{10}}{M_{00}} \qquad \bar{y} = \frac{M_{01}}{M_{00}}$$

En la documentación original en inglés puede obtenerse una guía de utilización de la aplicación así como los pasos necesarios para configurar el sistema para depurado.

4.1 Esquemático y Diagrama de Flujo




Sincronización e Inicialización de la Interfaz
Inicialización de la Kinect
Inicialización de la Kinect existosa?
Crear Hilo de Procesamiento de Imágenes
Desabilitar Menús de la Ventana
Mostrar Ventana de Aplicación
Gestionar Mensajes de la Cola de Ventanas
Petición de Finalización de la Aplicación?
Procesar Mensaje de Ventana
Destruir Ventana y Liberar Recursos